**Section 3 – Rendering Conditional Content and Lists**

Chapter 40 – Rendering Content Conditionally

1. v-if is used to add some condition in tag HTML using Vue.

<p v-if="goals.length === 0"> No goals have been added yet - please start adding some! </p>

Chapter 41 – v-if, v-else, and v-else-if

1. v-else and v-else-if only can be used directly after v-if, if there is some tag HTML between them, v-else and v-else-if can not be used.

Chapter 42 – Using v-show Instead Of v-if

1. v-show is used only for the tag HTML is stand alone, if you have multiple alternatives. The different between v-show and v-if is when using v-show if the condition is false, so the tag HTML will be hidden (display : none), but when using v-if the tag HTML will not be written (not display : none).
2. Using v-show is only for tag HTML which do a lot of status changes (visible or hidden).

Chapter 43 – Rendering Lists of Data

1. v-for is used to print all the data in Array or List (like foreach in JavaScript)

<ul v-show="goals.length > 0">

     <li v-for="goal in goals"> {{ goal }}</li>

</ul>

Chapter 44 – Diving Deeper Into v-for

1. We also can print index of foreach like below.

<ul v-show="goals.length > 0">

     <li v-for="(goal, index) in goals"> {{ goal }} - {{ index }}</li>

</ul>

1. We can also loop for object like below.

<ul>

<li v-for = "(value, key, index) in {name: 'Max', age:31}">

{{ key }} : {{ value }} - {{ index }}

</li>

</ul>

Chapter 45 – Removing List Items

1. We can remove one item in Array using Vue like below.

<ul v-show="goals.length > 0">

<li v-for = "(goal, index) in goals" @click="removeGoal(index)">

{{ goal }} - {{ index }}</li>

</ul>

Chapter 46 – Lists and Keys

1. @click.stop can be used to stop the listener, such as remove the item in Array. The code will be like below. We can add some input text, but there is a bug when you remove item in Array, the text which you typed in the input text will not suitable with your input before.

<ul v-show="goals.length > 0">

<li v-for="(goal, index) in goals" @click="removeGoal(index)">

      <p> {{ goal }} - {{ index }} </p>

      <input type="text" @click.stop>

</li>

</ul>

1. The solution for number 1 is add key = “goal”, goal must be very unique, so it can be differentiate one item with another item.

<ul v-show="goals.length > 0">

<li v-for="(goal, index) in goals" :key="goal" @click="removeGoal(index)">

<p> {{ goal }} - {{ index }} </p>

<input type="text" @click.stop>

</li>

</ul>

Summary

1. Conditional Content

* v-if (and v-show) allows you to render content only if a certain condition is met.
* v-if can be combined with v-else and v-else-if (only on direct sibling elements!)

1. Lists

* v-for can be used to render multiple elements dynamically.
* v-for can be used with arrays, objects, and ranges (numbers).

1. v-for Variations

* You can extract values, values and indexes or values, keys, and indexes.
* If you need v-for and v-if, DON’T use them on the same element. Use a wrapper with v-if instead.

1. Keys

* Vue re-uses DOM elements to optimize performance => This can lead to bugs if elements contain state
* Bind the key attribute to a unique value to help Vue identify elements that belong to list content.